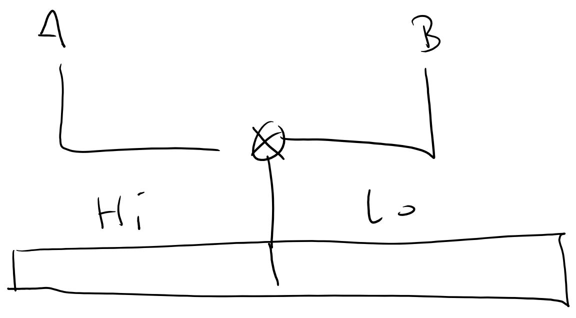
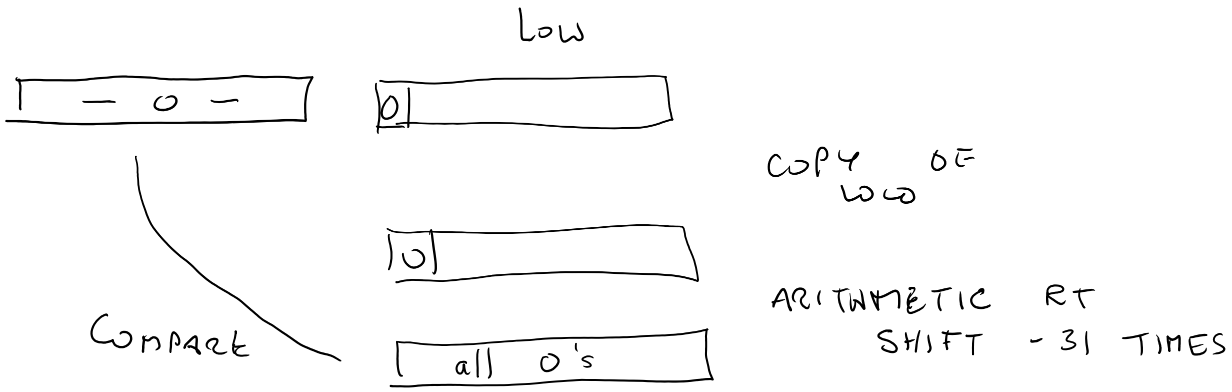
Lecture 8 - Pipelined Operations

Tuesday, February 18, 2020 10:40 AM

Objectives:	- Learn about how the MIPS processor uses a pipeline architecture to execute instructions - Learn about hazards that arise when using a pipeline architecture

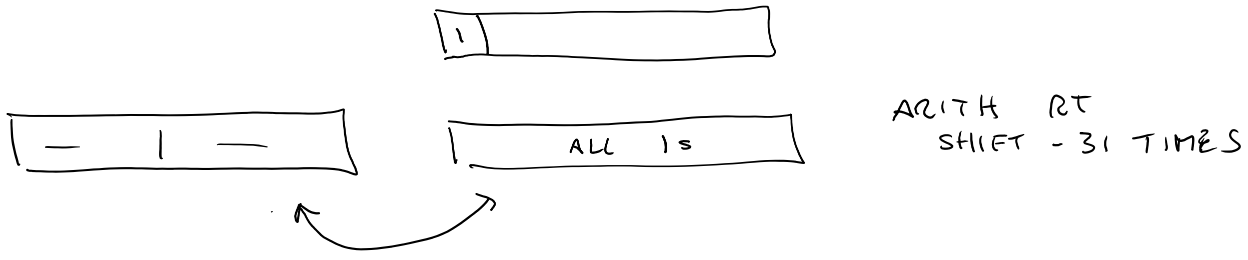


POSITIVE PRODUCT



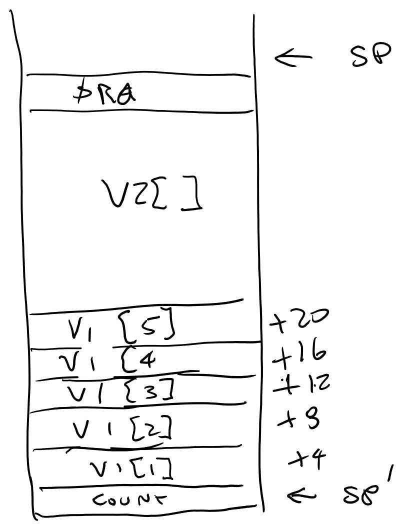
IF EQUAL
NO OVERFLOW

NEG PRODUCT



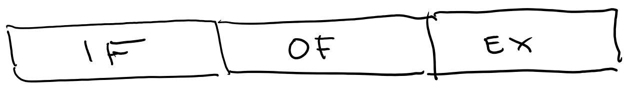
EQUAL → NO OVERFLOW

SETTING UP STACK



addi $sp, $sp, -48

FETCH, DECODE, EXECUTE



① INSTRUCTION FETCH -

LOAD INSTRUCTION FROM MEMORY
INTO IR
INCREMENT PC

② OPERAND FETCH

FETCH VALUES FROM REGISTERS
IF BRANCH INSTR CONDITION MET
UPDATE PC

③ EXECUTE - PERFORM ARITH. OPERATION
OR LOGIC OP

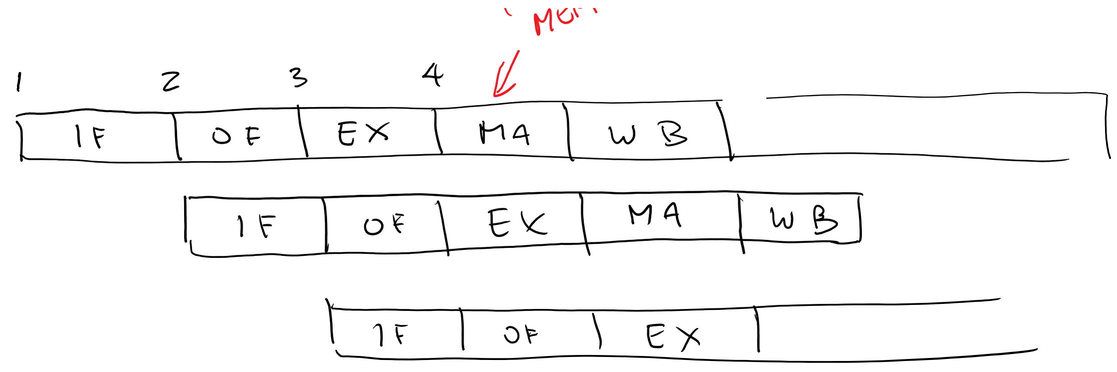
FOR LW & SW - CALCULATE EFFECTIVE ADDRESS

④ MEM ACCESS STAGE - IF LOAD, READ DATA FROM MEM
IF STORE, WRITE DATA TO MEM

ELSE PASS DATA FROM RESULT TO
WRITE BACK REG

⑤ WRITE BACK STAGE - STORE VALUE IN REGISTER

READ MEM



dest

addi $\bigcirc$t0, a0, 4 SOURCE

add t2, t0, t1

WITH MIPS NOT A PROBLEM DUE TO BYPASS

HAZARDS

Data Hazard

Lw \$t1, 0(\$t0)

addi \$a0, \$t1, 12

A VERY REAL HAZARD

NOP OR OTHER INSTR

CONTROL HAZARD

bgez t0, LOOP

addi t0, t0, 20

delay slot - INSTRUCTION FOLLOWING A BRANCH

branch delay - PC IS NOT UPDATED UNTIL AFTER THE NEXT INSTRUCTION IS ALREADY FETCHED.